IMPLEMENTASI FRAMEWORK FLASK PADA MODUL BETA-APP PADA APLIKASI SISTEM INFORMASI HELPDESK (SIH) STUDI KASUS PT XYZ

Jonathan Gea[1], Yeremia Alfa Susetyo[2]

[1,2]Pogram Studi Teknik Fakultas Teknologi Informasi Universitas Kristen Satya Wacana, Salatiga
e-mail : 672019111@student.uksw.edu[1], yeremia.alfa@uksw.edu[2]

ABSTRACT

PT XYZ is a large retail network in Indonesia that frequently receives reports of issues caused by the applications used within the company. To address this, a beta-app module was developed to enhance communication and coordination with reporters or relevant stores regarding the version of the application being used during these issues. The aim of this research is to ensure that application-related problems can be handled more effectively and efficiently, and the beta-app module can also serve as a means to obtain feedback on newly developed applications. The research follows the waterfall method in the software development process, resulting in an application that proves useful for PT XYZ in facilitating communication and coordination with reporters or stores. This application features the ability to create a beta-app, add and remove applications within the beta-app, and select branches and stores where the registered applications in the beta-app will be implemented. The architecture of the application utilized in this research includes a PostgreSQL database. The back-end is implemented using the Flask framework, a Python-based web development framework. The front-end is built using HTML, CSS, JavaScript, and Bootstrap. The entire system's functions have been tested using black box testing and have been proven to meet the expected requirements. This research contributes to the implementation of Flask, website development, and information systems, enabling PT XYZ to improve service quality and expedite the resolution process for application-related issues.

Keywords: *Flask Implementation, Website Development, Information Systems*

ABSTRAK

PT XYZ merupakan sebuah jaringan ritel besar di Indonesia yang sering menerima laporan kendala yang disebabkan oleh aplikasi yang digunakan di perusahaan. Oleh karena itu, dibangunlah modul *beta-app* untuk meningkatkan komunikasi dan koordinasi dengan pelapor atau toko terkait versi aplikasi yang digunakan saat terjadi kendala. Tujuan penelitian ini adalah untuk memastikan bahwa kendala terkait aplikasi dapat ditangani dengan lebih efektif dan efisien serta modul b*eta-app* juga dapat digunakan sebagai salah satu cara untuk mendapatkan *feedback* terkait aplikasi yang baru dikembangkan. Penelitian ini menggunakan metode *waterfall* dalam proses pengembangan perangkat lunak dan hasilnya adalah sebuah aplikasi yang berguna bagi PT XYZ untuk memudahkan komunikasi dan koordinasi dengan pelapor atau toko. Aplikasi ini memiliki fitur untuk membuat beta-app, menambahkan dan menghapus aplikasi di *beta-app*, serta memilih cabang dan toko di mana aplikasi yang terdaftar di beta-app yang akan diterapkan. Arsitektur aplikasi yang digunakan dalam penelitian ini meliputi *database PostgreSQL, Back-end* dengan *Framework Flask* dari

bahasa pemrograman *Python, front-end* dengan HTML, CSS, *JavaScript, dan Bootstrap*. Seluruh fungsi dalam sistem ini telah diuji dengan *Black box* testing dan terbukti sesuai dengan yang diharapkan. Penelitian ini dapat membantu PT XYZ untuk meningkatkan kualitas layanan dan mempercepat proses penyelesaian kendala terkait aplikasi.

Kata Kunci : Implementasi Flask, Pembangunan Website, Sistem Informasi

I. PENDAHULUAN

Pelaporan kendala atau helpdesk merupakan titik sentral untuk mengelola kendala atau masalah secara teratur dan terorganisir [1], di dukung perkembangan teknologi yang cepat telah mendorong manusia untuk menggunakan teknologi dalam segala aktivitasnya, termasuk di PT XYZ, sebuah jaringan ritel besar di Indonesia [2], tidak terkecuali dengan PT XYZ mengoperasikan lebih dari 15.000 cabang di seluruh Indonesia, dengan lebih dari 1.000 di antaranya merupakan gerai baru. Untuk memenuhi kebutuhan pasokan, perusahaan didukung oleh 32 pusat distribusi yang tersebar di 25 provinsi di Indonesia [3], PT XYZ mengembangkan aplikasi *Sistem Informasi Helpdesk* (SIH) untuk mencatat dan mengategorikan masalah serta solusinya. Aplikasi ini membantu perusahaan belajar dari pengalaman masa lalu, meningkatkan efisiensi operasional, kualitas layanan, kepuasan pelanggan, serta menjadi aset pengetahuan berharga untuk menghadapi tantangan di masa depan. [4].

Meskipun SIH telah membantu PT XYZ mengatasi banyak kendala, perusahaan masih sering menerima laporan kendala terkait aplikasi. Untuk menghindari kesalahpahaman tentang versi aplikasi yang digunakan, PT XYZ perlu meningkatkan komunikasi dengan pelapor atau toko terkait versi aplikasi yang digunakan. Solusinya adalah dengan membangun modul *beta*-app yang juga dapat digunakan untuk mendapatkan *feedback* terkait aplikasi baru yang dikembangkan. Dengan demikian, PT XYZ dapat menangani kendala aplikasi dengan lebih efektif dan efisien.

Dalam pengembangan aplikasi web, penggunaan Framework web seperti Python Flask mempercepat proses pengembangan. Penelitian ini menggunakan Python dan Flask untuk membangun aplikasi web [4]. Flask adalah web Framework yang ditulis dengan Python dan memudahkan pengaturan tampilan dan perilaku aplikasi web [5].

Oleh karena itu peneliti melakukan Implementasi *Framework Flask* Pada Modul *beta-app* Pada Aplikasi Sistem In*form*asi Helpdesk (SIH) Studi Kasus PT XYZ.

Dalam penelitian yang berjudul "Implementasi Flask Framework Pada Pembangunan Aplikasi Sistem Informasi

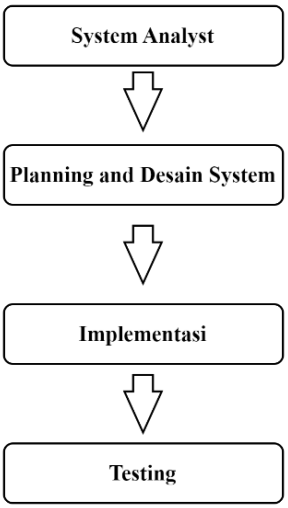
Helpdesk (SIH)", peneliti berhasil mengimplementasikan Flask Framework Python untuk membuat aplikasi SIH. Penelitian ini bertujuan untuk membangun aplikasi berbasis web yang digunakan sebagai langkah awal dalam pemecahan masalah jika terjadi kendala pada operasional perusahaan. Melalui aplikasi tersebut, setiap kendala yang dilaporkan juga dapat dimonitoring dengan mudah oleh user [4].

Selain itu, terdapat penelitian lain yang menggunakan Flask Framework dalam pengembangan aplikasi. Misalnya, penelitian yang berjudul "Implementasi Flask pada sistem penentuan minimal order untuk tiap item barang di distribution center PT XYZ berbasis website" [5]. dan "Pembangunan Aplikasi Virtual Inventory System (VIS) Berbasis Web Menggunakan Flask Framework (Studi Kasus: PT XYZ)" [6]. Kedua penelitian tersebut berhasil mengimplementasikan Flask Python untuk membuat aplikasi dengan tujuan yang berbeda.

Dengan demikian, penggunaan Flask Framework dalam pengembangan aplikasi web, termasuk dalam pembangunan modul beta-app pada aplikasi SIH PT XYZ, dapat memberikan manfaat dalam mempercepat proses pengembangan dan meningkatkan efisiensi operasional perusahaan.

II. METODE PENELITIAN

Penelitian ini menggunakan metode *waterfall*, sebuah model pengembangan perangkat lunak yang sering digunakan dalam proses pengembangan perangkat lunak. Metode *waterfall*, terdiri dari empat tahap utama: *system analyst*, planning dan desain system, implementasi, dan testing [7] seperti pada Gambar 1



Gambar 1. Research Flow

Berikut adalah penjabaran mengenai tahap-tahap metodologi penelitian yang digunakan oleh peneliti:

A. System Analyst

Tahap pertama dari metode Waterfall adalah System Analyst. Pada tahap ini, dil-akukan analisis kebutuhan *user* untuk menentukan spesifikasi fungsional dan nonfungsional dari perangkat lunak yang akan dikembangkan. Hal ini dilakukan dengan berkonsultasi langsung dengan

Programmer Analyst di PT XYZ untuk memahami kebutuhan terhadap sistem.

B. Planning and Design System

Tahap kedua dari metode Waterfall adalah Planning and Design System. Pada tahap ini, dilakukan perencanaan dan perancangan sistem yang mencakup perancangan arsitektur, antarmuka *user*, database, dan modul-modul sistem. Tujuan dari tahap ini adalah untuk memastikan bahwa desain sistem memenuhi kebutuhan *user* dan spesifikasi yang telah ditentukan.

C. Implementation

Tahap ketiga dari metode Waterfall adalah Implementasi. Pada tahap ini, dilakukan implementasi sistem dengan mengkodekan modul-modul sistem yang telah dirancang. Tahap ini sangat penting karena modul-modul sistem yang dibuat harus sesuai dengan spesifikasi yang telah ditetapkan dan dapat berfungsi dengan baik.

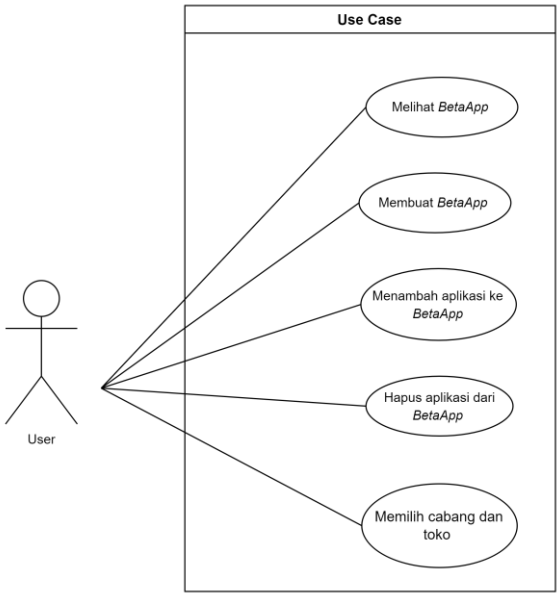
D. Testing

Tahap terakhir dari metode Waterfall adalah Testing. Pada tahap ini, dilakukan pengujian perangkat lunak untuk memastikan bahwa sistem yang telah dibuat memenuhi spesifikasi yang telah ditetapkan dan dapat berfungsi dengan baik.

III. HASIL DAN PEMBAHASAN

A. Perancangan dan Desain Sistem

Berdasarkan analisis sistem yang melibatkan wawancara dengan *Programmer Analyst* di PT XYZ mengenai kebutuhan sistem yang akan dibuat, peneliti menyimpulkan bahwa kebutuhan utama yang harus diterapkan pada modul *beta-app* tercantum dalam *use case* diagram berikut :

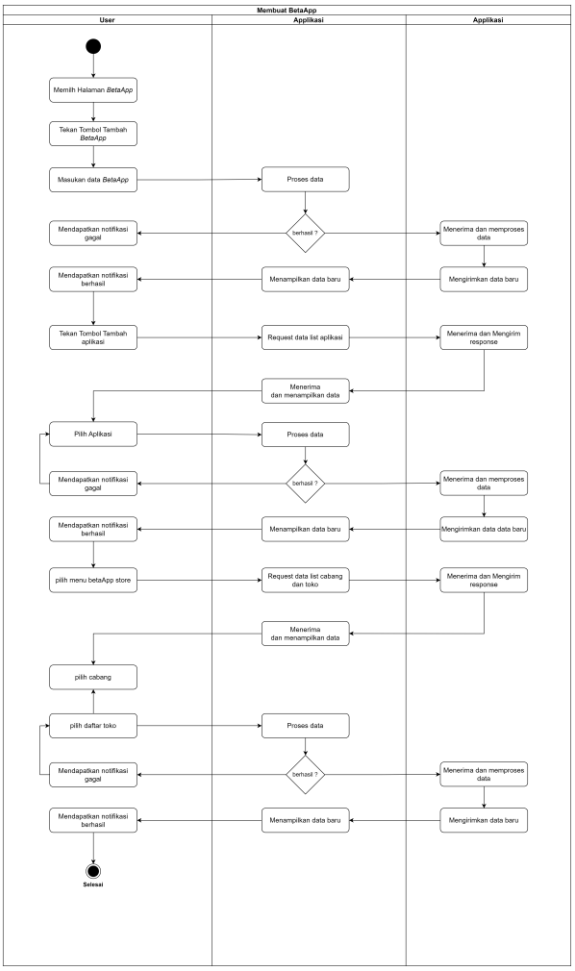


Gambar 2. Use case diagram beta-app

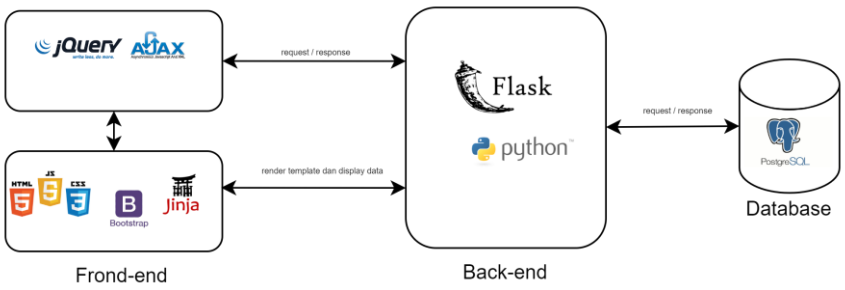
Gambar 2 menampilkan *use case diagram* yang menunjukkan pengunaan aplikasi oleh seorang *user*. Aplikasi ini memiliki beberapa fitur, di antaranya adalah membuat *beta-app* . Selain itu, *user* juga dapat menambahkan aplikasi ke *beta-app* yang sedang dalam percobaan dan menghapus aplikasi yang sudah didaftarkan di *beta-app* , fitur lainnya adalah *user* dapat memilih cabang dan toko di mana aplikasi

yang terdaftar di *beta-app* yang akan diterapkan. Dengan demikian, *user* dapat mengelola aplikasi yang sedang diuji coba dengan mudah dan efisien menggunakan aplikasi ini.

Diagram aktivitas beta-app digambarkan dalam gambar 3. Diagram tersebut menggambarkan langkah-langkah yang dilakukan oleh pengguna saat menggunakan aplikasi beta-app, mulai dari membuka halaman hingga menginput data, dan juga menggambarkan aktivitas sistem seperti menerima request dan mengirim respon, melakukan validasi dan proses data, serta berkomunikasi dengan database. Diagram ini juga dapat digunakan sebagai panduan oleh penulis dalam membangun aplikasi.



Gambar 3. *Activity diagram beta-app*



Gambar 4. Arsitektur sistem beta-app

Gambar 4 di atas merupakan arsitektur aplikasi yang digunakan dalam penelitian ini. Arsitektur ini terdiri dari beberapa komponen utama yang bekerja bersama-sama untuk menghasilkan aplikasi web yang berfungsi dengan baik. Komponen pertama adalah database *PostgreSQL* yang berperan dalam mengelola dan menyimpan data. Database ini digunakan untuk menyimpan berbagai data terkait aplikasi web.

Selain database, pada arsitektur aplikasi ini terdapat komponen *back-end* yang

berperan penting dalam mengolah data dan menghasilkan respon kepada *user*. *Back-end* dalam arsitektur ini dibangun dengan menggunakan *Framework Flask* dari bahasa pemrograman *Python*. Back-end ini berperan untuk memproses permintaan yang diterima dari *front-end* sebelum dikirim ke database.

Pada sisi *front-end*, arsitektur aplikasi ini menggunakan HTML, CSS, JavaScript, dan Bootstrap untuk memperindah tampilan website dan meningkatkan interaksi *user*. HTML digunakan untuk membangun struktur dan konten pada halaman web, sedangkan CSS digunakan untuk mengatur tampilan dan gaya pada halaman web. JavaScript digunakan untuk membuat halaman web lebih interaktif dan dinamis, sedangkan Bootstrap digunakan untuk memudahkan pengembangan tampilan pada aplikasi web dengan menyediakan komponen-komponen UI yang siap digunakan.

Selain itu, dalam pengembangan tampilan pada aplikasi web berbasis *Python*, arsitektur aplikasi ini menggunakan *Jinja* sebagai *template engine*. untuk meningkatkan interaksi antara *front-end* dan *back-end*, dalam pengembangan aplikasi web ini peneliti menggunakan teknologi AJAX (*Asynchronous JavaScript and XML*). AJAX memungkinkan aplikasi web untuk memperbarui data secara asinkron, sehingga memungkinkan *user* untuk tetap berada di halaman yang sama sambil data diperbarui [8]. Hal ini dapat meningkatkan pengalaman *user* dan efisiensi pengunaan aplikasi web. Teknologi AJAX diimplementasikan menggunakan *JavaScript* dan *jQuery*. *jQuery* dipilih untuk menyederhanakan penulisan kode *JavaScript* dan memiliki banyak fungsi yang dapat memudahkan pengembangan aplikasi web.

E. Implementasi Sistem

Flask sebagai *Framework* pada sisi back-end. Pada bagian awal code, terdapat import modul *Flask* pada baris ke-1 dan dilanjutkan dengan inisialisasi objek *Flask* pada baris ke-3. Objek ini berfungsi sebagai titik awal untuk membangun aplikasi *Flask*, kode program 1 digunakan untuk menjalankan *Flask* pada server. Pada baris ke-5, if statement digunakan untuk memastikan bahwa aplikasi dijalankan pada lingkungan yang tepat. Dalam hal ini, aplikasi *Flask* dijalankan pada lingkungan debug dengan host "localhost" dan port 8080 pada baris ke-6. seperti pada Kode Program 1.

```
1.  from Flask import Flask
2.
3. app = Flask(__name__)
4.
5. if __name__ == '__main__':
6.   app.run(debug=True,host="localhost",port=8080)
```

Gambar 5. Kode Program 1

Untuk menghubungkan sisi back-end dengan database PostgreSQL, digunakan library psycopg2. Library ini berfungsi sebagai interface untuk mengakses dan berinteraksi dengan database *PostgreSQL* melalui *Python*. Pada bagian kode program 2, dapat dilihat bahwa library psycopg2 diimport pada baris ke-1. Selanjutnya, pada baris ke-3 sampai ke-7, dilakukan koneksi dengan database PostgreSQL. Koneksi ini dilakukan dengan memasukkan parameter seperti host, database, *user*, dan password. Pada baris ke-3, objek koneksi (conn) dibuat dengan memanggil fungsi connect() dari library psycopg2. Objek koneksi ini berfungsi sebagai jembatan antara aplikasi *Flask* dengan database *PostgreSQL* yang digunakan. Setelah terhubung dengan database, maka aplikasi dapat melakukan berbagai macam operasi seperti memasukkan, mengambil, mengubah, dan menghapus data dari database., seperti pada kode program 2.

```
1. import psycopg2
2.
3. conn = psycopg2.connect(
4.     host="localhost",
5.     database="mydatabase",
6.     user="myusername",
7.     password="mypassword"
8. )
```

Gambar 6. Kode Program 2

Pada Kode Program 3, terdapat dua endpoint yang didefinisikan di sisi back-end. Endpoint pertama, '/' didefinisikan dengan decorator @app.route('/'). Endpoint ini akan diakses ketika *user* membuka halaman utama aplikasi. Pada endpoint ini, terdapat fungsi home() yang akan memproses data dan mengembalikan template 'layout.html' untuk ditampilkan ke *user*.

Sedangkan endpoint kedua, '/data' didefinisikan dengan decorator @app.route('/data'). Endpoint ini akan diakses ketika *user* mengirimkan data dari halaman *front-end*. Pada endpoint ini, terdapat fungsi data() yang akan memproses data yang diterima dan melakukan query ke database *PostgreSQL* untuk mengambil data dari tabel. Jika data berhasil diambil, fungsi akan mengembalikan respon dalam bentuk JSON menggunakan fungsi jsonify_response() yang akan mengirimkan status code 200 sesuai dengan aturan kode status http [9] dan data yang diambil dari tabel sebagai respons. Namun, jika data tidak ditemukan atau gagal diambil, fungsi akan mengembalikan respon dengan status code 400 [9] dan pesan error .

Dalam endpoint '/data', terdapat juga pengunaan fungsi *request.form.get("data")* untuk mengambil data yang dikirimkan dari halaman *front-end*. Fungsi ini akan mengembalikan nilai *None* jika data tidak ditemukan. Jika nilai yang dikembalikan bukan *None*, maka akan dilakukan query ke database *PostgreSQL*.

```
1. from Flask import render_template, request, jsonify
2.
3. @app.route('/')
```

```
4. def home():
5.     # proses data
6.     render_template("layout.html")
7.
8. @app.route('/data')
9. def data():
10.     # proses data
11.     data = request.form.get ("data")
12.     if data is not None :
13.         cursor = conn.cursor()
14.         cursor.execute("SELECT * FROM table_data")
15.         result = cursor.fetchall()
16.         return jsonify_response(200, 'Success',
'OK', result)
17.     return jsonify_response(400, 'Error', 'Data
Requred Faild')
```

Gambar 7. Kode Program 3

Kode Program 4 adalah fungsi *json-ify_response()* yang digunakan untuk menghasilkan respon dalam bentuk JSON yang lebih rapi dan terstruktur. Fungsi ini menerima parameter status code, status, pe-san, dan data. Data yang dikirimkan akan diatur ulang menjadi *form*at JSON dengan menggunakan library jsonify sebelum dikirimkan ke *front-end*.

Pada route '/data', terdapat proses un-tuk memeriksa apakah data telah diterima. Jika sudah, maka akan dilakukan koneksi ke database dan melakukan *query* untuk mengambil data dari tabel. Setelah itu, data yang diambil akan dikirim ke *front-end* menggunakan fungsi jsonify_response yang menggunakan library jsonify agar data yang dikirimkan lebih rapi seperti pada Kode Program 4.

```
1. def jsonify_response(status_code,status, mes-
sage=None, data=None):
2.     response = {
3.         'status_code': status_code,
4.         'status': status,
5.         'message': message,
6.     }
7.     if data:
8.         response['data'] = data
9.     return jsonify(response)
10.
```

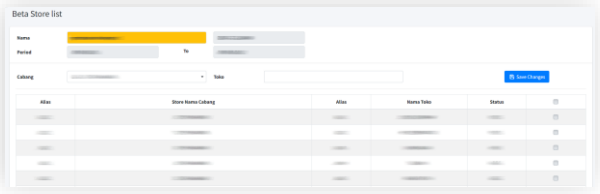
Gambar 8. Kode Program 4

Terakhir, untuk mengirim *request* dan menerima *response* dari *back-end*, digunakan teknologi AJAX Seperti pada Kode Program 5 dibawah ini.

```
1. $.ajax({
2.     type: 'PUT',
3.     url: "{{ url_for ('data') }}",
4.     data: formData,
5.     beforeSend: function () {
6.         showLoading();
7.     },
8.     success: function (response) {
9.         hideLoading()
10.         if (response.status_code = 200){
11.                 msg('Pesan berhasil terkirim!',
'success');
12.             displaydata()
13.         }if else (response.status_code = 400){
14.             msg('response.message', 'warning');
15.         }
16.     },
17.     error: function (xhr, status, error) {
18.         alert('Terjadi kesalahan!');
19.         console.log(xhr.responseText);
20.     }
21. });
22.
```

Gambar 9. Kode Program 5

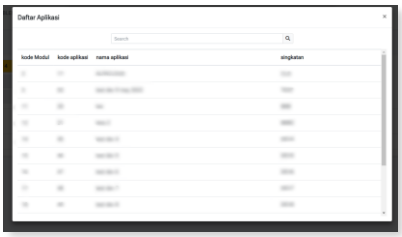
F. Tampilan Aplikasi



Gambar 10. Halaman beta-app

Gambar 10 menunjukkan tampilan halaman *beta-app* . Di halaman ini, *user* dapat membuat aplikasi beta dengan menekan tombol "Tambah" di sudut kanan atas. Setelah itu, akan muncul modal pop-up yang memungkinkan *user* mengimputkan data nama dan periode *beta-app* . Selain itu, *user* juga dapat menambahkan aplikasi ke *beta-app* dengan menekan tombol "Tambah Aplikasi" di sudut kanan atas. Setelah itu, akan muncul modal pop-up yang berisi daftar aplikasi

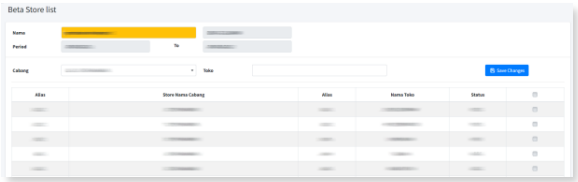
seperti pada gambar 11 dan data setiap aplikasi. Setelah *user* memilih aplikasi yang diinginkan, aplikasi tersebut akan ditambahkan ke daftar aplikasi *beta-app* . *User* juga dapat menghapus aplikasi dengan menekan tombol "Hapus" di kolom aksi.



Gambar 11. modal pop-up daftar aplikasi

User juga dapat melihat detail *beta-app* dengan menekan kolom berwarna kuning. Setelah itu, akan muncul daftar *beta-app* beserta datanya, seperti pada gambar 12. Setelah *user* memilih *beta-app* yang diinginkan, data *beta-app* akan terisi secara otomatis di halaman dan *user* dapat menambah dan menghapus aplikasi yang terdaftar di *beta-app* .

Gambar 12. modal top-up daftar beta-app

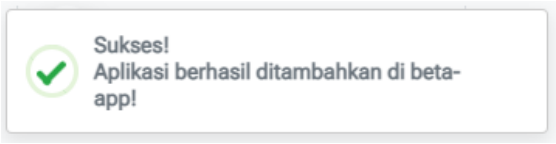


Gambar 13. Halaman beta-app store

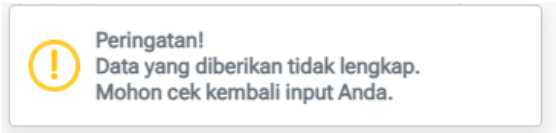
Gambar 13 menampilkan tampilan halaman *beta-app Store*, di mana *user* dapat memilih cabang dan toko di mana aplikasi yang terdaftar di *beta-app* akan diterapkan. Pertama-tama, *user* memilih *beta-app* dengan menekan kolom berwarna kuning. Setelah itu, *user* dapat memilih cabang dengan menekan inputan cabang, dan dropdown list cabang akan muncul. Setelah cabang dipilih, sistem akan menampilkan daftar toko sesuai dengan cabang yang dipilih. Selanjutnya, *user* dapat memilih toko dengan menekan checkbox pada kolom sebelah kanan. Jika *user* ingin memilih semua toko, *user* dapat menekan checkbox yang berada di header table, sehingga semua toko akan terpilih secara otomatis. Setelah selesai memilih toko, *user* dapat menekan tombol Simpan Perubahan di bagian kanan atas tabel.

Sistem juga menampilkan alert pop-up di atas layar, beberapa contoh seperti yang terlihat pada Gambar 14. Alert sukses menunjukkan bahwa proses telah berhasil dilakukan, Gambar 15 menunjukkan alert peringatan yang muncul ketika validasi *front-end* telah dimodifikasi oleh *user*. Sistem melakukan validasi di sisi server dan mengembalikan respon peringatan. Untuk mengurangi kesalahan *user* untuk Tindakan yang berbahaya [10], sistem juga memberikan pesan konfirmasi sebelum men-

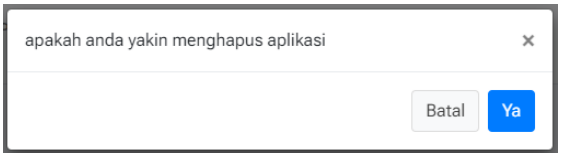
jalankan tindakan yang diminta *user*. Contoh pesan konfirmasi dapat dilihat pada Gambar 16. Jika *user* yakin dengan perintah yang diberikan, *user* dapat menekan tombol "Ya" untuk menjalankan proses. Namun, jika *user* tidak yakin atau terjadi kesalahan tekan, *user* dapat menekan tombol "Batal" untuk membatalkan atau tidak melanjutkan proses tersebut."



Gambar 14. alert Sukses



Gambar 15. alert peringatan



Gambar 16. alert konfirmasi

G. *Testing*

Adapun pelaksanaan pengujian pada sistem dilakukan dengan menggunakan teknik pengujian *black box* testing salah satu strategi pengujian yang dikenal sebagai data driven testing atau input/output testing[11] . *Black box* testing adalah teknik pengujian perangkat lunak yang fokus pada spesifikasi fungsional, mengabaikan struktur kontrol, dan memungkinkan pengembang membuat himpunan kondisi input yang melatih syarat fungsional program. Keuntungan metode ini adalah penguji tidak perlu pengetahuan bahasa pemrograman tertentu, pengujian dilakukan dari sudut pandang *user* [12].

Tabel 1. hasil testing aplikasi.

Langkah Pengujian	hasil yang diharapkan	status
Melihat *Beta-app*		
user masuk kehalaman *beta-app* , lalu menekan kolom kuning	sistem menampilkan modal pop-up dan memberikan daftar *beta-app*	Ok
user memilih *beta-app*	Sistem memberikan notifikasi bahwa pengisian data pada halaman *beta-app* berhasil dan data seperti nama, periode, *user*, NIK, dan daftar aplikasi yang sudah ditambahkan di	Ok

	beta-app akan terisi secara otomatis pada halaman *beta-app* .	
Membuat *Beta-app*		
user masuk kehalaman *beta-app* , lalu menekan tombol "tambah"	sistem menampilkan modal pop-up *form* pengisi data *Beta-app*	Ok
user mengisi *form*	*form* dapat diisi	Ok
user mengisi *form* tanggal	muncul pop-up tanggal dan tanggal dapat diisi dengan *form*at yang benar	Ok
user menekan tombil "simpan" tanpa mengisi data dengan lengkap	Sistem memberikan notifikasi bahwa ada *form* yang belum terisi	Ok
user menekan tombil "simpan" setelah mengisi data dengan lengkap	Sistem memberikan notifikasi bahwa pembuatan b*eta-app* berhasil dan data seperti nama, periode, *user*, NIK, dan daftar aplikasi yang sudah ditambahkan di *Beta-app* akan terisi secara otomatis pada halaman *Beta-app* .	Ok
Menambah aplikasi ke *Beta-app*		
setelah *user* masuk kehalaman dan memilih *beta-app* , *user* menekan tombol "tambah aplikasi"	sistem menampilkan modal pop-up *form* pengisian data aplikasi	Ok
user menekan tombol kolom kuning dengan label "aplikasi"	sistem menampilkan daftar aplikasi	Ok

user mimilih aplikasi	sistem menggisi *form* pengisian data aplikasi secara otomatis	Ok
user mengisi *form*	*form* dapat diisi	Ok
user menekan tombol "Simpan" setelah mengisi data dengan lengkap	Sistem memberikan notifikasi bahwa aplikasi berhasil terdaftar di *Beta-app* dan aplikasi tersebut terdaftar di *Beta-app* .	Ok
user menekan tombol "Simpan" tanpa mengisi data dengan lengkap	Sistem memberikan notifikasi bahwa ada *form* yang belum terisi	Ok
Menghapus aplikasi dari *Beta-app*		
setelah *user* masuk kehalaman dan memilih *beta-app* , *user* memilih aplikasi yang mau dihapus dan menekan tombol "hapus" dikolom aksi	sistem memberikan notifikasi alert, apakah *user* yakin menghapus aplikasi dari *beta-app*	Ok
user menekan tombol batal	Notifikasi ditutup dan tidak terjadi perubahan atau penghapusan aplikasi.	Ok
user menekan tombol yakin	Sistem memberikan notifikasi bahwa aplikasi berhasil dihapus, dan aplikasi terhapus	Ok
Memilih cabang dan toko		
user masuk kehalaman *beta-app* store, lalu menekan kolom kuning	sistem menampilkan modal pop-up dan memberikan daftar *beta-app*	Ok
user memilih *beta-app*	Sistem memberikan notifikasi bahwa pengisian data	Ok

	pada halaman *beta-app* berhasil dan data seperti nama, periode, *user*, NIK, dan daftar aplikasi yang sudah ditambahkan di *beta-app* akan terisi secara otomatis pada halaman *beta-app* .	
user menekan dropdwon inputan cabang	dropdown akan terbuka dan sistem menampilkan opsi-opsi cabang yang tersedia	Ok
user melakkukan pencarian cabang dengan menuliskan nama cabang	sistem memberikan data sesuai dengan text yang diberikan *user*	Ok
user memilih cabang	sistem menampilkan toko berdasarkan cabang	Ok
user memilih checkbox, untuk memilih toko	Centang pada kotak (checkbox) telah terisi.	Ok
menghapus tanda centang pada kotak (checkbox) untuk membatalkan pendaftaran toko.	Centang pada kotak (checkbox) menjadi kosong.	Ok
user menekan tombol simpan setelah *user* melakukan perubahan pada daftar toko	sistem memberikan notifikasi alert, apakah *user* yakin menyimpan	Ok
user menekan tombol batal	Notifikasi ditutup dan tidak terjadi perubahan pada daftar toko	Ok
user menekan tombol yakin	Sistem memberikan notifikasi bahwa daftar toko berhasil diupdate	Ok

user menekan tombol simpan setelah tanpan melakukan perubahan pada daftar toko	muncul notifikasi bahwa *user* belum melakkukan perubahan	Ok

Tabel 1 menunjukkan hasil pengujian, di mana status "OK" menandakan bahwa hasil output pengujian telah sesuai. Sementara itu, jika statusnya "NOK", maka menunjukkan adanya kesalahan yang perlu diperbaiki pada fungsi tersebut. Setelah Penelit melakukan pengujian terhadap seluruh fungsi dalam sistem ini dan semua fungsi telah sesuai, maka sistem ini siap digunakan oleh *user*.

IV. SIMPULAN

Hasil penelitian implementasi *Framework Flask* pada modul *beta-app* pada aplikasi sistem in*form*asi helpdesk (SIH) menghasilkan sebuah aplikasi yang berguna bagi PT XYZ untuk memudahkan komunikasi dan koordinasi dengan pelapor atau toko, terkait versi aplikasi yang digunakan saat terjadi kendala di dalam perusahaan. Dalam penelitian ini, *Framework Flask* digunakan oleh peneliti dengan bahasa pemrograman *Python* karena lebih ringan dan mudah diimplementasikan. Dalam penelitian ini, peneliti berhasil menyesuaikan *Framework* dengan kebutuhan aplikasi dengan menambahkan library dari pihak ketiga. Berdasarkan hasil pengujian sistem menggunakan metode Black Box, semua fungsi pada sistem telah sesuai sehingga aplikasi ini dapat digunakan oleh *user*. Saran dari penulis adalah, penelitian selanjutnya sebaiknya mempertimbangkan penggunaan metode pengembangan perangkat lunak yang lebih adaptif dan fleksibel, seperti metode Agile, guna memungkinkan perubahan dan penyesuaian yang lebih cepat terhadap kebutuhan yang berkembang. Selain itu, dalam hal pengembangan front-end, penulis sangat merekomendasikan penggunaan React.js. Implementasi React.js akan membantu meningkatkan pengalaman pengguna dan mempercepat waktu pengembangan.

DAFTAR PUSTAKA

[1] W. Likhar and H. Purwanto, "ANALISA DAN PERANCANGAN SISTEM INFORMASI TICKETING HELPDESK ONLINE BERBASIS WEB: STUDI KASUS PT XYZ." [Online]. Available:

http://www.help-desk-world.com/help-

[2] Kidi, "TEKNOLOGI DAN AKTIVITAS DALAM KEHIDUPAN MANUSIA (sebuah tinjauan)," Nusa Tenggara Barat, 2018. Accessed: Nov. 01, 2022. [Online]. Available: https://bpsdmd.ntbprov.go.id/wp-content/uploads/2018/05/Teknologi-dan-aktivitas-dalam-kehidupan-manusia.pdf

[3] "PT Sumber Alfaria Trijaya, Tbk. dan entitas anaknya and its subsidiaries." [Online]. Available: www.alfamart.co.id

[4] Candra Wijayanto and Yeremia Alfa Susetyo, "Implementasi Flask Framework Pada Pembangunan Aplikasi Sistem Informasi Helpdesk (SIH)," *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)*, vol. 07, no. 03, pp. 858–868, 2022, [Online]. Available: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi/article/view/3161/1328

[5] G. F. Novindri, P. Ocsa, and N. Saian, "IMPLEMENTASI FLASK PADA SISTEM PENENTUAN MINIMAL ORDER UNTUK TIAP ITEM BARANG DI DISTRIBUTION CENTER PADA PT XYZ BERBASIS WEBSITE," 2022.

[6] E. Haezer, Y. Kristianto, and N. Setiyawati, "PEMBANGUNAN APLIKASI VIRTUAL INVENTORY SYSTEM (VIS) BERBASIS WEB MENGGUNAKAN FLASK FRAMEWORK (STUDI KASUS: PT XYZ)," 2021.

[7] M. Muslih et all, "Implementasi Metode Waterfall Dalam Pembangunan Sistem Informasi Klinik Tiara Bunda Berbasis Web Service," *Jurnal Rekayasa Teknologi Nusa Putra*, vol. 5, no. 2, pp. 20–25, 2019.

[8] H. S. Tampake, "Penggunaan AJAX pada Pengembangan Aplikasi Web," *Jurnal Teknologi Informasi-Aiti*, vol. 4, no. 1, pp. 1–100, 2007.

[9] "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content," RFC 7231. Accessed: Apr. 01, 2023. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7231#section-3.1.1.1

[10] Jakob Nielsen, "Confirmation Dialogs Can Prevent User Errors — If Not Overused," Nielsen Norman Group. Accessed: Mar. 31, 2023. [Online]. Available:

https://www.nngroup.com/articles/confirmation-dialog/#:~:text=Let's%20go%20back%20to%20basics,click%20Yes%20without%20thinking%20further.

[11] Glenford J. Myers, Corey Sandler, and Tom Badgett, *114-the-art-of-software-testing-3-edition*. 2011.

[12] T. Snadhika Jaya, P. Studi Manajemen Informatika, J. Ekonomi dan Bisnis, and P. Negeri Lampung JlnSoekarno, "Pengujian Aplikasi dengan Metode Blackbox Testing Boundary Value Analysis (Studi Kasus: Kantor Digital Politeknik Negeri Lampung)," *Jurnal Informatika: Jurnal Pengembangan IT (JPIT)*, vol. 03, no. 02, 2018.